无监督异常检测在自动驾驶竞速模仿学习中的应用：深度解析与复现指南

1. 绪论：自动驾驶中的数据卫生挑战

在自动驾驶技术的快速演进中，模仿学习（Imitation Learning, IL）已成为一种核心范式，特别是在端到端（End-to-End）的视觉控制任务中。与依赖复杂手工规则或难以定义的奖励函数的强化学习（Reinforcement Learning, RL）不同，模仿学习允许智能体通过观察专家演示来直接学习策略。这种方法在自动驾驶竞速（Autonomous Racing）领域尤为盛行，因为人类驾驶员能够直观地处理复杂的赛道动态，而无需明确建立物理模型[1]。

然而，模仿学习的有效性存在一个根本性的假设前提：训练数据的质量必须极高。具体而言，专家演示不仅要求驾驶行为本身是完美的，还要求传感器数据（如摄像头输入的图像）是清晰且无噪声的。在实际的物理世界部署中，这一假设经常被打破。数据集中往往充斥着“脏数据”或异常样本，例如摄像头镜头上的雨滴、碰撞导致的视角遮挡、或是由于光照变化产生的眩光[1]。如果训练数据中包含这些异常样本，基于行为克隆（Behavioral Cloning, BC）的神经网络往往会错误地将这些视觉伪影与特定的驾驶动作（如急转弯或刹车）关联起来，导致所谓的“因果混淆”（Causal Confusion）。这不仅会降低模型的泛化能力，更会导致车辆在实际运行中出现灾难性的失控[1]。

传统的解决方案依赖于人工清洗数据，但这对于长达数小时甚至数天的高频驾驶视频来说，成本过于高昂且效率极低。另一种方案是使用半监督异常检测，但这需要预先构建一个纯净的“正常”数据集作为基准，这在面对海量未标记的原始数据时同样陷入了“鸡生蛋，蛋生鸡”的困境[1]。因此，如何在没有任何标签的情况下，直接从混合了正常和异常样本的原始数据集中自动识别并剔除异常帧，成为了提升模仿学习鲁棒性的关键瓶颈。

本文将深入剖析佛罗里达大学（University of Florida）Trustworthy Engineered Autonomy (TEA) 实验室在IROS 2025上发表的研究成果《Unsupervised Anomaly Detection Improves Imitation Learning for Autonomous Racing》。该研究提出了一种基于卷积自编码器（Convolutional Autoencoder, CAE）和新型“潜在参考损失”（Latent Reference Loss）的无监督异常检测框架，能够在不需要任何人工标签的情况下，自动识别并过滤掉包括碰撞、遮挡和环境干扰在内的多种异常数据[1]。我们将从理论基础、算法原理、实验验证到代码复现的每一个细节进行详尽的拆解，旨在不仅让读者理解其核心机制，更能具备复现并改进这一系统的能力。

2. 理论基础与问题形式化

2.1 模仿学习与行为克隆的数学本质

在自动驾驶竞速场景中，我们的目标是训练一个神经控制器 $h : \mathcal{X} \to \mathcal{A}$，其中 $\mathcal{X}$ 代表高维的传感器输入空间（例如，160×120 像素的RGB图像），$\mathcal{A}$ 代表连续的动作空间（通常包括转向

角度和油门值）[1]。行为克隆是模仿学习中最直接的形式，它将这一问题视为一个监督学习问题。

给定一个包含 N 个样本的训练数据集 $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$，其中 x_i 是第 i 帧图像，y_i 是对应的人类专家操作。行为克隆的目标是最小化预测动作与专家动作之间的差异，通常采用均方误差（MSE）作为损失函数：

$$\mathcal{L}_{BC}(\theta) = \frac{1}{N}\sum_{i=1}^{N} ||y_i - h_\theta(x_i)||_2^2$$

其中 θ 是网络的参数。然而，这一公式隐含地假设了所有的 (x_i, y_i) 对都是从专家策略的理想分布中采样的。现实情况是，数据集 $\mathcal{D}$ 实际上是由两部分组成的混合体：高质量的正常数据 $\mathcal{D}_{clean}$ 和低质量的异常数据 $\mathcal{D}_{dirty}$，即 $\mathcal{D} = \mathcal{D}_{clean} \cup \mathcal{D}_{dirty}$ [1]。

如果模型在包含 $\mathcal{D}_{dirty}$ 的全集上训练，神经网络强大的拟合能力会使其学习到异常图像特征与动作之间的错误映射。例如，如果包含“撞墙”画面的数据通常伴随着“左转”指令（因为驾驶员试图从墙边驶离），网络可能会在看到墙壁纹理时自动触发左转，即使在不需要左转的直道上也是如此。

2.2 无监督异常检测的挑战

本研究的核心任务是学习一个置信度函数 $f(x_i, y_i) = Pr$，在没有任何监督信号（即不知道哪些是脏数据）的情况下，识别出 $\mathcal{D}_{dirty}$ 中的样本[1]。

这一设定与传统的“单类分类”（One-Class Classification, OCC）不同。OCC通常假设训练数据是纯净的，仅测试数据包含异常。而在本研究的设定中，训练数据本身就是被污染的。这就要求检测算法具有极强的鲁棒性：它必须能够利用正常数据在统计上的主导地位（假设 $|\mathcal{D}_{clean}| \gg |\mathcal{D}_{dirty}|$），主动“遗忘”或无法重构那些稀疏分布的异常样本[1]。

3. 基于潜在参考损失的重构异常检测

为了解决上述问题，研究团队提出了一种改进的卷积自编码器（CAE）架构。传统的自编码器通过编码器 $e_\psi(x)$ 将输入压缩为低维潜在向量 h，再通过解码器 $d_\phi(h)$ 将其重构为 x'。异常检测的直觉是：由于模型主要在正常数据上训练，它应该能很好地重构正常样本，而无法有效重构异常样本。因此，重构误差（Reconstruction Error）可以作为异常分数的代理。

然而，深度神经网络往往具有过强的泛化能力，甚至能够很好地重构异常图像，导致检测失效。为了打破这种“过度泛化”，该研究引入了核心创新点——潜在参考损失（Latent Reference Loss）[1]。

3.1 核心机制：潜在参考损失 ($\mathcal{L}_{refer}$)

潜在参考损失的设计基于一个几何假设：在潜在空间（Latent Space）中，正常样本的分布是密集的，而异常样本是稀疏且远离正常簇的。如果在训练过程中，强制要求每个样本的潜在表示 h_i 必须接近某个“参考样本”的潜在表示，那么由于正常样本占据大多数，异常样本更有可能被强制拉向正常样本的簇，从而丢失其自身的异常特征信息[1]。

具体实现步骤如下：

1. 参考集构建：在训练开始前，从原始数据集 $\mathcal{D}$ 中随机采样 M 个样本（其中 $M < N$），构成参考数据集 $\mathcal{D}_{refer}$。由于 $\mathcal{D}$ 中绝大多数是正常样本，$\mathcal{D}_{refer}$ 也将主要由正常样本组成[1]。
2. 潜在空间映射：在每一轮训练迭代中，将 $\mathcal{D}_{refer}$ 中的图像通过当前的编码器，得到参考潜在特征集 $\mathcal{H}_{refer} = \{h_1^r, ..., h_M^r\}$。
3. 最近邻搜索：对于当前批次（Batch）中的每一个输入图像 x_i，计算其潜在向量 h_i，并在 $\mathcal{H}_{refer}$ 中找到与其欧氏距离最近的邻居 h_i^{near}：

$$h_i^{near} = \arg\min_{h_j^r \in \mathcal{H}_{refer}} ||h_i - h_j^r||_2$$

[1]

4. 损失计算：定义潜在参考损失 $\mathcal{L}_{refer}$ 为 h_i 与其最近邻 h_i^{near} 之间的均方误差：

$$\mathcal{L}_{refer} = \frac{1}{N}\sum_{i=1}^{N} ||h_i - h_i^{near}||_2$$

[1]

3.2 总体优化目标

CAE的总损失函数 $\mathcal{L}$ 由像素级的重构损失 $\mathcal{L}_{rec}$ 和潜在参考损失 $\mathcal{L}_{refer}$ 组成，通过超参数 λ 进行加权：

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda \mathcal{L}_{refer}$$

[1]

其中重构损失通常采用均方误差：

$$\mathcal{L}_{rec} = \frac{1}{N}\sum_{i=1}^{N} ||x_i - d_\phi(e_\psi(x_i))||_2$$

1

工作原理解析：

这两个损失函数在训练过程中存在一种博弈关系。$\mathcal{L}_{rec}$ 试图让解码器完美还原输入图像，这要求 h_i 保留尽可能多的输入细节（包括异常细节）。相反，$\mathcal{L}_{refer}$ 试图让 h_i 变得像参考集中的某个样本（大概率是正常样本）。

- 对于正常样本，它在 $\mathcal{D}_{refer}$ 中很容易找到相似的邻居，因此 $\mathcal{L}_{refer}$ 和 $\mathcal{L}_{rec}$ 是一致的，模型能学到高质量的重构。
- 对于异常样本（如镜头上有水滴），它在 $\mathcal{D}_{refer}$ 中的最近邻通常是一个没有水滴的正常图像。最小化 $\mathcal{L}_{refer}$ 会强迫编码器忽略“水滴”这一特征，将潜在向量 h_i 拉向“无水滴”的状态。结果，解码器根据这个被“净化”的潜在向量重建图像时，会生成一个没有水滴的正常图像。
- 最终，异常输入的原始图像（有水滴）与重构图像（无水滴）之间会产生巨大的结构差异，从而由于高重构误差而被检测出来[1]。

3.3 异常度量与自适应阈值

为了量化原始图像与重构图像之间的差异，研究团队没有单纯使用MSE，而是采用了皮尔逊相关系数（**Pearson Correlation Coefficient, PCC**）。PCC对光照变化不敏感，更关注图像的结构相似性（Structural Similarity），这对于检测诸如雨滴或透明遮挡物等结构性异常更为有效[1]。

PCC r_i 的计算公式为：

$$r_i = \frac{\sum(x_i - \bar{x}_i)(x'_i - \bar{x}'_i)}{\sqrt{\sum(x_i - \bar{x}_i)^2 \sum(x'_i - \bar{x}'_i)^2}}$$

1

其中 x_i 和 x'_i 分别是原始图像和重构图像，$\bar{x}$ 表示像素均值。PCC值越接近1，表示重构越完美（正常）；值越低，表示差异越大（异常）。

为了处理赛道不同路段（直道vs弯道）带来的自然重构波动，系统对PCC序列应用了窗口大小为100的中值滤波器（**Median Filter**）。最终的异常判定阈值 τ 是自适应确定的：

$$\delta = median(r_{all}) - 0.05$$

[1] 任何平滑后PCC值低于 δ 的帧都被标记为异常并从训练集中剔除[1]。

4. 实验平台与网络架构详解

为了复现该研究，深入了解其硬件平台和网络架构的具体参数至关重要。

4.1 DonkeyCar 自动驾驶平台

该研究是在 **DonkeyCar S1** 平台上进行的。DonkeyCar 是一个开源的 DIY 自动驾驶小车项目，广泛用于学术研究和教学。

- 底盘：1/16 或 1/10 比例的遥控车（RC Car）。
- 计算单元：通常配备 Raspberry Pi 4 或 NVIDIA Jetson Nano。为了运行深度学习模型，Jetson Nano 提供了必要的 GPU 加速能力[10]。
- 传感器：单目广角摄像头。虽然具体型号可能变化，但研究中明确指出输入图像的分辨率被缩放为 **160x120** 像素[6]。
- 控制接口：通过 PCA9685 伺服驱动板控制转向舵机和驱动电机（ESC）[11]。

4.2 网络架构参数

根据文献描述，系统包含两个独立的网络：用于清洗数据的CAE和用于控制车辆的CNN控制器。以下是推断出的详细层级结构[1]：

表 1: 卷积自编码器 (CAE) 架构

[1]

模块	层级类型	滤波器数量/神经元	卷积核大小	激活函数	备注
Encoder	Conv2D	16	3x3	ReLU	提取低级特征
	Conv2D	32	3x3	ReLU	提取中级特征
	Conv2D	64	3x3	ReLU	提取高级特征
	Flatten	-	-	-	展平特征图

	Dense (FC)	256	-	ReLU	潜在空间瓶颈层
Decoder	Dense (FC)	(H/8 * W/8 * 64)	-	ReLU	映射回特征图大小
	Reshape	-	-	-	重塑为张量
	Conv2DTranspose	64	3x3	ReLU	上采样
	Conv2DTranspose	32	3x3	ReLU	上采样
	Conv2DTranspose	16	3x3	Sigmoid	输出层，归一化到

(注：具体的输入尺寸为160x120，因此经3次下采样后特征图尺寸约为20x15)

表 2: 模仿学习控制器 (CNN Controller) 架构

[1]

层级类型	滤波器数量/神经元	卷积核大小	步长 (Stride)	激活函数	备注
Conv2D	24	5x5	2x2	ReLU	类似NVIDIA DAVE-2
MaxPool	-	2x2	-	-	空间下采样
Conv2D	32	3x3	2x2	ReLU	特征提取
Conv2D	64	3x3	2x2	ReLU	特征提取
Flatten	-	-	-	-	
Dense	128	-	-	ReLU	全连接层

Dense	64	-	-	ReLU	全连接层
Dense	1	-	-	Tanh	输出转向角 [-1, 1]

4.3 训练超参数

- 学习率 **(Learning Rate)**: CAE 使用 0.0005，CNN 控制器使用 5×10^{-5} [1]。
- 批次大小 **(Batch Size)**: CAE 为 256，CNN 控制器为 512 [1]。
- 优化器: 均为 Adam [1]。
- 训练轮数 **(Epochs)**:
  - 对于“雨滴”和“塑料遮挡”数据集，CAE训练 100 Epochs。
  - 对于“撞墙”数据集，CAE仅训练 10 Epochs（防止过拟合明显的撞墙特征）[1]。
  - CNN控制器训练 100 Epochs [1]。
- λ **(**损失权重**)**: 设为 1 [1]。

5. 实验结果与性能分析

该研究通过在DonkeyCar平台上收集的三种真实异常场景——碰撞（**Wall Hit**）、透明遮挡（**Plastic Bag**）、雨滴（**Raindrop**）——验证了方法的有效性。数据集包含了约16,000帧正常图像和1,600帧异常图像（比例10:1）[1]。

5.1 异常检测精度

与最先进的无监督视频异常检测方法 Generative Cooperative Learning (GCL) 相比，本文提出的方法在检测召回率（Recall）和精度（Precision）上均取得了显著优势，特别是在难以察觉的“雨滴”和“塑料遮挡”场景中。

表 3: 异常检测性能对比 **(Recall / Precision)**

[1]

方法	碰撞 **(Wall Hit)**	雨滴 **(Raindrop)**	塑料遮挡 **(Plastic)**
CAE + $\mathcal{L}_{refer}$ **(Ours)**	**0.964 / 1.000**	**1.000 / 1.000**	**1.000 / 0.924**
GCL (SOTA)	1.000 / 0.972	0.188 / 0.108	0.035 / 0.067

CAE Only (Ablation)	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000

数据表明，普通的CAE（Ablation）完全失效，因为它能够完美重构异常图像，导致无法通过重构误差区分异常。GCL虽然能检测到明显的撞墙，但在细微干扰下表现极差。而本文方法几乎实现了完美的检测[1]。

5.2 赛道性能提升：横向偏差 (CTE)

最终的评价指标是小车在赛道上自主行驶时的横向偏差（**Cross-Track Error, CTE**）。CTE越小，说明车辆行驶越稳定，越接近中心线。通过在天花板安装摄像头进行外部位置追踪，研究测量了不同模型控制下的CTE[1]。

实验结果显示，使用经过清洗的数据集训练的模型，其CTE相比基线模型（使用全数据训练）降低了 **25%** 到 **40%**[1]。

- 雨滴场景: CTE 从 0.135 降至 0.089。
- 塑料遮挡场景: CTE 从 0.117 降至 0.072。
- 理想状态 **(Oracle)**: 使用完全纯净数据训练的CTE约为 0.063，说明本文方法的效果已非常接近理论上限[1]。

6. 复现指南：从代码到部署

本节将指导您如何一步步复现该论文的实验。我们将假设您拥有基本的Linux环境和Python编程能力。

6.1 环境搭建

首先，您需要准备硬件和软件环境。

- 硬件: 建议使用带有 NVIDIA GPU 的工作站进行训练（如 RTX 3060 或更高），以及一个 DonkeyCar 实体车（或使用 Donkey Gym 模拟器）。
- 代码库: 论文代码已在 TEA Lab 的 GitHub 仓库开源。

```
Bash
# 克隆 TEA Lab 的 DonkeyCar 分支
git clone https://github.com/Trustworthy-Engineered-Autonomy-Lab/DonkeyCar-TEA
# 或者官方 DonkeyCar 库（如果只需要基础功能）
git clone https://github.com/autorope/donkeycar
```

14

6.2 数据采集与预处理

如果您无法进行实体车实验，建议使用 **Donkey Gym** 模拟器生成数据。

1. 正常数据采集: 在模拟器中手动驾驶或使用预训练模型跑 10-20 圈，收集约 16,000 张图像

(tub 数据格式)。

2. 异常注入:
   - 模拟雨滴: 编写一个 Python 脚本，使用 OpenCV 在图像上随机位置叠加半透明的白色或灰色圆形遮罩。
   - 模拟遮挡: 对图像应用局部高斯模糊或添加噪声块。
   - 模拟撞墙: 专门录制一段车辆贴着墙壁行驶或撞击墙壁的视频，截取约 1,600 帧。
3. 数据集划分: 确保数据集中包含 10:1 的正常与异常数据，不要打乱顺序，以便观察时序特性，但在训练CAE时需要随机打乱[1]。

6.3 实现潜在参考损失 (Latent Reference Loss)

这是复现的核心难点。在 Keras 或 PyTorch 中自定义 Loss 函数。以下是概念代码逻辑:

Python

```
# 伪代码逻辑展示 Latent Reference Loss 的实现
import tensorflow as tf

class LatentReferenceLoss(tf.keras.losses.Loss):
    def __init__(self, reference_latent_set, lambda_weight=1.0):
        super().__init__()
        self.ref_set = tf.constant(reference_latent_set) # 预先计算好的参考集潜在向量
        self.lambda_weight = lambda_weight

    def call(self, y_true, y_pred):
        # y_true 是原始图像, y_pred 是重构图像
        # 注意:这里需要修改模型架构，使其输出不仅包含重构图，还包含潜在向量 h

        # 1. 计算重构损失 (MSE 或 L2)
        rec_loss = tf.reduce_mean(tf.square(y_true - y_pred_image))

        # 2. 计算潜在参考损失
        # 对 batch 中的每个 h，在 ref_set 中找最近邻
        # 这是一个计算密集型操作，通常使用矩阵运算加速
        distances = tf.norm(current_batch_h[:, None, :] - self.ref_set[None, :, :], axis=-1)
        min_distances = tf.reduce_min(distances, axis=1)
        ref_loss = tf.reduce_mean(min_distances)

        return rec_loss + self.lambda_weight * ref_loss
```

关键步骤：

1. 在每个 Epoch 开始前，从训练集中随机抽取 $M = 1024$ 张图像作为参考集 $\mathcal{D}_{refer}$ 。
2. 通过编码器推理得到 $\mathcal{H}_{refer}$ 。
3. 在训练循环中，计算当前 Batch 的潜在向量与 $\mathcal{H}_{refer}$ 的距离矩阵，取最小值作为 Reference Loss [1]。

6.4 异常检测与数据清洗

训练完成后，使用训练好的 CAE 对整个数据集进行推理：

1. 计算每张图片的重构图。
2. 计算原始图与重构图的 PCC 值。
3. 对 PCC 曲线应用 window_size=100 的中值滤波。
4. 计算阈值 $\delta = \text{median}(PCC_{smoothed}) - 0.05$ 。
5. 过滤掉所有 $PCC < \delta$ 的样本[1]。

6.5 训练控制器与评估

使用清洗后的数据集（仅包含 x 和 y），训练标准的 DonkeyCar CNN 模型（categorical 或 linear 输出）。在测试阶段，计算 CTE 需要建立世界坐标系。如果您在模拟器中，可以直接读取 pos_x, pos_y 和赛道中心线坐标进行欧氏距离计算。如果在实体环境中，则需复现论文中的“天花板摄像头”定位系统，这需要额外的计算机视觉工程（如颜色阈值分割追踪绿色标记 ）[1]。

7. 结论与展望

本研究通过引入潜在参考损失，成功解决了一般自编码器在异常检测中“过度泛化”的难题，为自动驾驶数据清洗提供了一种高效、无监督的解决方案。其25-40%的性能提升证明了数据质量在模仿学习中的决定性作用。

未来的工作可能会探索多模态融合（如结合LiDAR和雷达数据），利用不同传感器之间的潜在空间一致性来进一步提高检测的鲁棒性。此外，动态阈值调整机制也有望解决从室内到室外光照剧烈变化时的适应性问题[15]。对于致力于构建可信赖自动驾驶系统的研究者而言，这种无需标注的“自我净化”能力将是通向更高安全性的重要阶梯。

引用说明：本文档中的数据、图表描述及理论引用均基于用户提供的研究片段（Snippets），主要对应文献 [1] (IROS 2025 Paper) 及相关辅助材料 [2] 等。代码复现路径参考了 TEA Lab 和 DonkeyCar 官方开源库的结构。

引用的著作

1. Unsupervised_Anomaly_Detection_Improves_Imitation_Learning_for_Autonomous_Racing.pdf
2. Unsupervised Anomaly Detection Improves Imitation Learning for Autonomous Racing | PDF - Slideshare, 访问时间为 二月 3, 2026, https://www.slideshare.net/slideshow/unsupervised-anomaly-detection-improves-imitation-learning-for-autonomous-racing/283887883
3. IROS 2025 Program | Wednesday October 22, 2025, 访问时间为 二月 3, 2026, https://ras.papercept.net/conferences/conferences/IROS25/program/IROS25_ContentListWeb_2.html
4. Unsupervised Anomaly Detection Improves Imitation Learning for Autonomous Racing - IEEE Xplore, 访问时间为 二月 3, 2026, https://ieeexplore.ieee.org/document/11246113
5. Self-Trained Deep Ordinal Regression for End-to-End Video Anomaly Detection | Request PDF - ResearchGate, 访问时间为 二月 3, 2026, https://www.researchgate.net/publication/343461578_Self-Trained_Deep_Ordinal_Regression_for_End-to-End_Video_Anomaly_Detection
6. Vision-based racing car stops crashing after unsupervised data cleaning, 访问时间为 二月 3, 2026, https://www.youtube.com/watch?v=RjJ3nZR6_RQ
7. Networking Systems for Video Anomaly Detection: A Tutorial and Survey - GitHub, 访问时间为 二月 3, 2026, https://github.com/fdjingliu/NSVAD
8. A comprehensive review on deep learning-based methods for video anomaly detection, 访问时间为 二月 3, 2026, https://www.researchgate.net/publication/347241990_A_comprehensive_review_on_deep_learning-based_methods_for_video_anomaly_detection
9. Image encoding selection based on Pearson correlation coefficient for time series anomaly detection - ResearchGate, 访问时间为 二月 3, 2026, https://www.researchgate.net/publication/375175502_Image_encoding_selection_based_on_Pearson_correlation_coefficient_for_time_series_anomaly_detection
10. Install the software. - Donkey Car, 访问时间为 二月 3, 2026, https://docs.donkeycar.com/guide/install_software/
11. Install Software on Donkey Car – City Tech | Autonomous Vehicles Race, 访问时间为 二月 3, 2026, https://openlab.citytech.cuny.edu/avrace/getting-started/install-software-on-donkey-car/
12. From Simulated to Real Test Environments for Self-Driving Cars - Brian Pulfer, 访问时间为 二月 3, 2026, https://www.brianpulfer.ch/docs/Brian%20Pulfer%20-%20From%20Simulated%20to%20Real%20Test%20Environments%20for%20Self%20Driving%20Cars.pdf
13. Create an autopilot. - Donkey Car, 访问时间为 二月 3, 2026, https://docs.donkeycar.com/guide/train_autopilot/
14. Trustworthy Engineered Autonomy Lab - GitHub, 访问时间为 二月 3, 2026, https://github.com/Trustworthy-Engineered-Autonomy-Lab
15. Unsupervised Anomaly Detection for Autonomous Robots via Mahalanobis SVDD with Audio-IMU Fusion - arXiv, 访问时间为 二月 3, 2026, https://arxiv.org/html/2505.05811v1

16. daniel-bogdoll/phd: Anomaly Detection for Autonomous Driving - GitHub, 访问时间为 二月 3, 2026, https://github.com/daniel-bogdoll/phd